New Model for Membrane Peeling in Ocular Surgery Simulator

N. Mukai[a], M. Harada[a], K. Muroi[a], T. Hikichi[b], and A. Yoshida[b]

[a] *Information Technology R&D Center, Mitsubishi Electric Corporation, Kamakura, Japan (mukai@isl.melco.co.jp)*

[b] *Department of Opthalmology, Asahikawa Medical College, Asahikawa, Japan (hikichi@asahikawa-med.ac.jp)*

Abstract: We have developed a vitreous surgery simulator, with which ophthalmologists can train for vitreous surgery without mentors. It can provide stereo visual and force feedback during the surgery training. The simulator is very useful because some pathological cases happen only in human eyes and there is no alternative training method. In the simulator, a disease called pre-retinal membrane is implemented, in which case a membrane proliferates on the retina. The main purpose of the simulator is to learn how to peel the membrane. Therefore, it is very important to simulate the membrane peeling process in real-time. To simulate physical phenomena, the mass-spring model is usually used; however the model is very time-consuming to calculate the realistic surgery images in real-time. In this paper, we propose a new membrane-peeling model that calculates only the important points for the membrane deformation and interpolates others by using calculated points. With this model, we could simulate a realistic membrane peeling process in real-time.

Keywords: Surgical simulators; Computer graphics; Virtual reality; Mass-spring model; Haptic devices

1. INTRODUCTION

Computer graphics, virtual reality, real-time simulation, and some related technologies have improved rapidly. These technologies are combined and applied to several fields such as industrial, medical, and entertainment. Especially the medical field; virtual reality based surgical simulators are important because surgeons need enough training opportunities in order to improve their surgery skills. With these backgrounds, several medical surgery simulators have been developed [Suzuki et al,. 1998a and Neumann et al., 1998] in addition to the vitreous surgery simulator [Mukai et al. 1999a&b] The purpose of the simulator is for the training of vitreous surgery and it is very useful for ophthalmologists because some ocular diseases happen only in human eyes and there is no alternative training method even with the usage of animal bodies. With the simulator, ophthalmologists can train for vitreous surgery repeatedly without using eyes of various animals. They can improve their surgery skills by themselves so that their mentors are not tied down teaching them.

The Simulator uses many virtual reality technologies. Operators get a stereo view of a virtual surgery process as visual feedback through the binoculars, which mimic an ocular surgery microscope. They use two surgical instruments with both hands; one is a light guide that illuminates the inside of an eye and the other is an operation instrument such as a vitreous cutter, peeler, and forceps. They feel friction that occurs between the eye and the instruments when they insert them into the eye. Also, there are two foot-switches. One is a surgery microscope pedal, by which trainees can change microscope position, focus, zoom ratio, and so forth. The other is a vitreous cutter pedal, by which they can control the power of core vitrectomy that extracts vitreous humor using vitreous cutter.

The main purpose of the simulator is to learn how to peel the membrane. The membrane is like a thin soft small paper, which sticks to the retina. If trainees touch the retina with surgical instruments, bleeding happens. The bleeding size is designed to change depending on the distance between the retina and the tip position of the instrument, which penetrates into the retina. Surgeons can learn how to peel the membrane without causing bleeding.

In order to simulate physical phenomena, several models are used such as the mass-spring model [Takahashi et al., 1996] and the sphere model [Suzuki et al,. 1998b]. However, these models are too time-consuming to simulate surgery process in real-time, because these models need many points to construct the object shape and it takes a long time to calculate every point position which deforms by giving a force from outside. Therefore, we propose a new model that is different from mass-spring or sphere models and that can simulate realistic surgery process in real-time.

2. ARCHITECTURE OF NEW MODEL

2.1 Membrane Model

In order to simulate an object deformation in real-time, which is usually defined as 30 Hz or 30 frames/s for visual feedback, it is necessary to reduce the quantity of the points that construct an object and it reduces the quality of the object image. Therefore, we introduce a new model, which calculates only important points constructing an object and interpolates others by using calculated important points. In this new model, membranes are defined with two kinds of important point: outline and inner boundary point.

Figure 1 illustrates one example of a membrane model. In the figure, an eye is modeled as a semi-sphere that has a radius R, and solid and dotted lines show the outline and the inner boundary of the membrane respectively. In the inside of the inner boundary, every point sticks to the retina Each line is composed of several points, which can be mapped onto grid points (large black and white points) so that the outline and the inner boundary can be redefined by drawing lines which connect two grid points of the outline (large black points) and the inner boundary (large white points) respectively, and then,

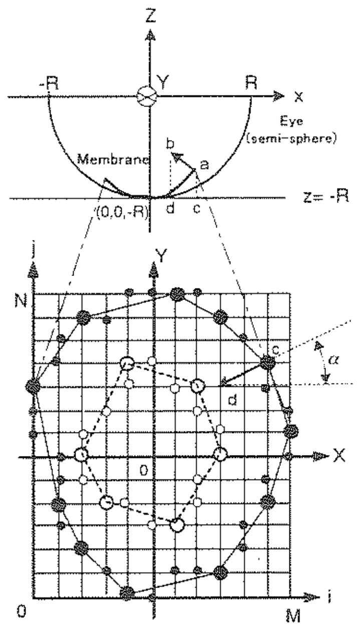

Figure 1. Membrane model mapped onto a grid.

these lines are digitized and some intermediate points are created (small black and white points). Suppose that X and Y dimensions of the grid are M and N respectively, the number of the outline pointsis $OutNum$ which includes both big and small black points, and X and Y coordinates of the outline points are stored in $Out.X$ and $Out.Y$ respectively in the counterclockwise order. Also, we introduce functions $Cx(k)$ and $Cy(k)$ which translate the outline point index k into X and Y grid coordinates of the point respectively, and $FLAG$ which indicates the state of the point: IN , OUT or $EDGE$. Then, the inner judge algorithm, which judges which point belongs to the inside of the inner boundary (IN), the outside of the boundary (OUT) or on the boundary ($EDGE$), is the following (Figure 2).

In Figure 2, after every point is initialized as OUT , only the outline point is marked as $EDGE$. First, check the directions of the outline points from index k to $k+1$. If the direction from index $k-1$ to k is reverse to the one of the above directions, the outline point k is a protruding point so that there is no need to search. Otherwise, search for inside points in the perpendicular direction to the other direction and mark them as IN until $EDGE$ is found. As a result, points that have IN or $EDGE$ are in the inside or on the edge of the inner boundary respectively and they make the membrane planes.

```
/* Initialize every point as outside* /
for (i = 0; i < M; i + +)
  for (j = 0; j < N; j + +)
    FLAG(i, j) = OUT;
/* Initialize outline points as EDGE* /
for (k = 0; k < OutNum; k + +)
  FLAG(Cx(k), Cy(k)) = EDGE;
/* Prepare for searching* /
Out.X(OutNum) = Out.X(0); Out.Y(OutNum) = Out.Y(0);
/* Search inner points* /
for (k = 1; k < = OutNum; k + +){
  i = Cx(k); j = Cy(k);
  /* search -Y direction* /
  if (Out.X(k + 1) < Out.X(k) && Out.Y(k + 1) < = Out.Y(k)){
    if (Out.X(k) > Out.X(k - 1) && Out.Y(k) > Out.Y(k - 1))
      break; /* no need to search* /
      while (FLAG(i, j - 1) = = EDGE){
        j = j - 1; FLAG(i, j) = IN; } }
  /* search +Y direction* /
 if (Out.X(k + 1) > Out.X(k) ) && Out.Y(k + 1) >= Out.Y(k){
   if (Out.X(k) < Out.X(k - 1) ) && Out.Y(k) < Out.Y(k - 1)
     break; /* no need to search* /
     while (FLAG(i, j + 1) == EDGE){
       j = j + 1; FLAG(i, j) = IN; } }
  /* search - X direction* /
  if (Out.Y(k + 1) > Out.Y(k) ) && Out.X(k + 1) <= Out.X(k){
   if (Out.X(k) > Out.X(k - 1) ) && Out.Y(k) < Out.Y(k - 1)
     break; /* no need to search* /
     while (FLAG(i - 1, j) == EDGE){
      i = i - 1; FLAG(i, j) = IN; } }
  /* search + X direction* /
  if (Out.Y(k + 1) < Out.Y(k) ) && Out.X(k + 1) >= Out.X(k){
   if (Out.X(k) < Out.X(k - 1) ) && Out.Y(k) > Out.Y(k - 1)
     break; /* no need to search* /
     while (FLAG(i + 1, j) == EDGE){
      i = i + 1; FLAG(i, j) = IN; } }
  }
}
```

Figure 2. Inner judge algorithm.

2.2 Deformation Method

In this new model, only the outline and the inner boundary points are calculated for the deformation, and the rest of the points are interpolated by using calculated points. In this section, the calculation method of these important points is discussed. The above algorithm can be applied to both convex and concave polygons; however, only convex polygons are considered in order to simplify the discussion.

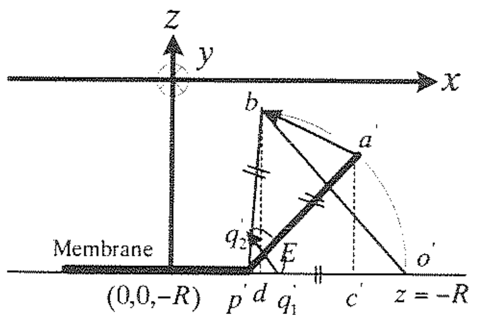

Figure 3. Calculation of an inner boundary point.

In the case of pre-retinal membrane, the membrane is a thin soft small and flat plane, and located at the center of the eyeground so that we can create the membrane model as in Figure 3. In the figure, the origin of the Cartesian coordinate (x, y, z) is placed at the center of the eye and the membrane is placed around $(0,0,-R)$. Also as in Figure 1, the membrane has the grid coordinate (i, j), the outline (black points), and the inner boundary (white points). Suppose that we pick up a point $a(a_x, a_y, a_z)$, which is one of the outline points, and move it to another point $b(b_x, b_y, b_z)$ along $\overrightarrow{ab}$. Here, we introduce a plane $z = -R$ and project a and b onto this plane so that new points $c(a_x, a_y, -R)$ and $d(b_x, b_y, -R)$ are obtained. Consider a plane which includes $\overrightarrow{ab}$ and $\overrightarrow{cd}$, and rotate the plane by $-\alpha$ around $\overrightarrow{db}$ so that y coordinate of every point on this plane becomes b_y, where $\cos\alpha = |a_x - b_x| / |\overrightarrow{cd}|$ (Figure 1). Then, the plane is transformed as in Figure 3, and new points, $a'(a_x', b_y, a_z)$ and $c'(a_x', b_y, -R)$, are obtained. The line passing d and c' is defined as follows.

$$\frac{x - b_x}{a_x' - b_x} = t(parameter) \tag{1}$$

Now we search a point $p'(x, b_y, -R)$, which is on the above line and has the same distance from both a' and b (Figure 3). Point p' satisfies the following condition.

$$(x - a_x')^2 + (-R - a_z)^2 = (x - b_x)^2 + (-R - b_z)^2 \tag{2}$$

From (1) and (2), we can solve the parameter t and get the point p, with the rotation of p' by α around

$\overrightarrow{db}$. Here, the model supposes that the membrane can be peeled with p as the center of the rotation. If p is in the outside of the inner boundary, we can peel the membrane without any forces from the membrane; however, if p is in the inside of the inner boundary, there is a force from the membrane, which sticks to the retina. Then, we have to calculate the updated inner boundary by taking the force from the membrane into account. Suppose that the force F_m from the membrane is defined as follows.

$$\begin{aligned} F_m &\propto D \quad if\ D \le E \\ &= 0 \quad\ \ if\ D > E \end{aligned} \tag{3}$$

Where, D: Distance between point b and it's initial position o' (Figure 3)
E: Elastic limit

With Figure 3, let us calculate the point o', which is the initial position of b and on the plane $z = -R$. From Figure 3, $|o'p'| = |bp'|$. Then o' can be written as follows.

$$o' = (p'_x + |bp'|, b_y, -R) \tag{4}$$

Then, consider two points q'_1 and q'_2, which have the following conditions.
a) q'_1 is on the line $o'p'$.
b) q'_2 is on the line bp'.
c) Distance between q'_1 and q'_2 is E.
d) $\overrightarrow{q'_1q'_2}$ is parallel to $\overrightarrow{o'b}$.

The point q'_1 is solved as follows.

$$q'_1 = (p'_x + \frac{E}{|o'b|}|o'p'|, b_y, -R) \tag{5}$$

Thus, with the rotation of q'_1 by α around $\overrightarrow{db}$, the new inner boundary point $q(q_x, q_y, q_z)$ is obtained. Supposed that the membrane is peeled uniformly, the updated inner boundary is calculated as follows. First, calculate a line L, which passes the point q and crosses $\overrightarrow{cd}$ perpendicularly (Figure 4).

$$\overrightarrow{cd} = (b_x - a_x, b_y - a_y, 0) \tag{6}$$

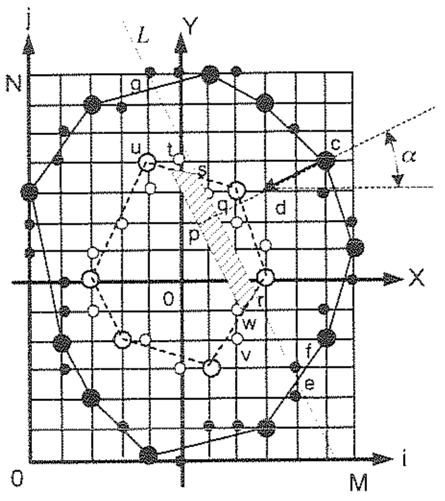

Figure 4. Update of Inner boundary points.

Therefore, the line L can be written as follows.

$$\frac{x - q_x}{-(b_y - a_y)} = \frac{y - q_y}{b_x - a_x} = t \tag{7}$$

Secondly, calculate the intersection points of the line L and the inner boundary, which is translated onto the plane $z = -R$. As a result, two intersection points r and s are calculated (Figure 4). Then, r and s are translated onto the grid and z coordinate is calculated from the eye model because the inner boundary sticks to the retina. Finally, the new updated boundary $rstu...vw$ is constructed. With the same method, another line, which passes p and crosses $\overrightarrow{cd}$ perpendicularly, and the area of the hatched part in Figure 4 are calculated. The area is considered to create the force from the membrane.

Also, the outline points, which should move by the peeling, are calculated: as $ef...c...g$ in Figure 4.The membrane peeling action is like a rotation around the line L and the rotation matrix M is calculated with five points a, b, c, q, and h as the following (Figure 5).

$$\cos\theta = \frac{|aq|^2 + |bq|^2 - |ab|^2}{2|aq||bq|} \tag{8}$$

$$\cos\phi = \frac{|cq|^2 + |hq|^2 - |ch|^2}{2|cq||hq|} \tag{9}$$

$Where, c(a_x, a_y, -R), h(a_x, q_y, -R)$

$$M = M_t M_z M_y M_z^{-1} M_t^{-1} \tag{10}$$

$$M_t = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ q_x & q_y & 0 & 1 \end{pmatrix}$$

$$M_z = \begin{pmatrix} \cos\phi & \sin\phi & 0 & 0 \\ -\sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$M_y = \begin{pmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

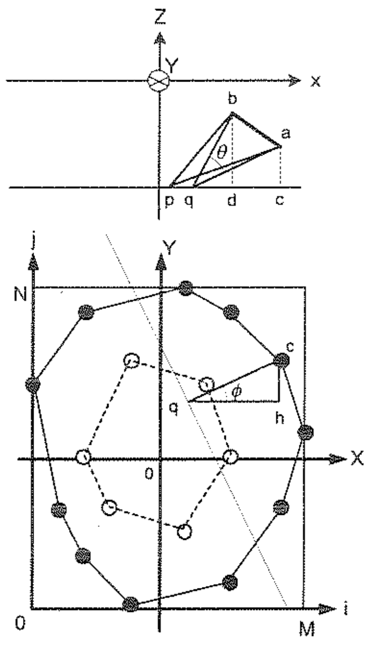

Figure 5. Calculation of rotation matrix.

According to (10), the outline points are translated by $(-q_x, -q_y, 0)$ with M_t^{-1} so that q becomes $(0,0,0)$. They are rotated by $-\phi$ around z axis with M_z^{-1} so that $\overrightarrow{cq}$ is on xz plane, and they are rotated by θ around y axis with M_y so that a reaches b. Then, with the reverse transformation with $M_t M_z$, the new outline points, which are rotated by θ around the line L, are obtained.

2.3 Real-time Deformation

In the above section, the new updated outline and the inner boundary points have been calculated. The real time membrane deformation algorithm is shown in Figure 6.The outline ($EDGE$) and the inner boundaries ($BOUND$) are important points and calculated precisely so that their values should not be changed. On the other hand, other points (IN) are not important and interpolated by using these important points. First, the membrane area is searched and the grid area is expanded by one for four directions (up, down, right and left) so that every point $P(x, y, z)$ has four neighborhoods. $ALLOWANCE$ is the threshold error of this algorithm. If the error is over the $ALLOWANCE$, the algorithm stops.

```
/* Search the membrane area* /
/* Calculate just once at the beginning* /
minI = M; maxI = 0;
minJ = N; maxJ = 0;
for(j = 0; j < N + 1; j + + ){
  for(i = 0; i < M + 1; i + + ){
    if (FLAG(i, j) == EDGE or FLAG(i, j) == IN){
      if (i < minI) minI = i;
      if (i > maxI) maxI = i;
      if (j < minJ) minJ = j;
      if (j > maxJ) maxJ = j;
    }
  }
}

/* Real time deformation* /
do{
  error = 0;
  for(j = minJ; j < maxJ + 1; j + + ){
    for(i = minI; i < maxI + 1; i + + ){
      if (FLAG(i, j) == IN && FLAG(i, j)! = EDGE)
      && FLAG(i, j)! = BOUND
{
        OLD = P(i, j);
        P(i, j) = (P(i, j - 1) + P(i, j + 1)) + P(i - 1, j) + P(i + 1, j)/4;
        if (distance(OLD, P(i, j)) > error)
        error = distance(OLD, P(i, j));
      }
    }
  }
} while(error > ALLOWANCE);
```

Figure 6. Real-time deformation algorithm.

3. RESULTS

The simulator uses Dell Precision 610 (Pentium III 550MHz 2CPUs 512MB Memories) with Intense3D WildCat4000 as the graphics accelerator to create the stereo images of the surgery. Figure 7 shows the membrane peeling image with this new model. In the experiment, the grid size of the membrane ($M \times N$) was 128×128 and the membrane was constructed with 945 polygons. The frame rate is 30Hz so that the membrane deforms very smoothly. On the other hand, the membrane created with the mass-spring model had to be reduced to 103 polygons in order to simulate the peeling process in real-time (30Hz). In general, the image quality of the membrane depends on the amount of the polygons which construct the membrane so that the membrane image created with this new model is much better than that with mass-spring model.

In Figure 7, the left hand instrument is called a light-guide and it illuminates the eyeground. The right hand instrument is called a peeler and the trainee is trying to peel the membrane with this instrument. By illuminating the eyeground with the light-guide, the shadow of the peeler is shown near the optic nerve head (ONH), which is a light circle on the right side of the image. If the trainee touches the eyeground with the surgical instrument accidentally, bleeding happens. In Figure 7, two bleedings happen; one is near ONH and the other is near the light-guide. The size of the bleeding is designed to depend on the length between the eyeground and the tip position of the surgical instrument which penetrates into the eyeground.

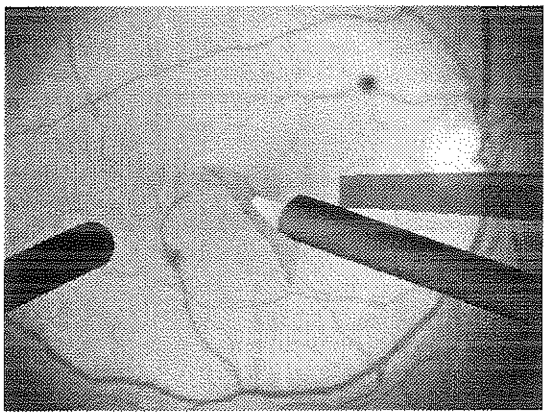

Figure 7. Membrane Peeling Process.

4. CONCLUSIONS

In this paper, the new model for the membrane peeling is described In this model, only the important points, which are the outline of the membrane and the inner boundary of the membrane, is calculated precisely and the other points which construct the membrane are interpolated by using calculated important points. With this new model, we could develop the vitreous surgery simulator, with which trainees can train for vitreous surgery and improve their skills. The system has been evaluated by some ophthalmologists and obtains a reputation that the membrane peeling process with this model simulates real vitreous surgeries very well.

5. ACKNOWLEDGEMENTS

We would like to express our special thanks to Mr. Terada, Mr. Miyamoto, Mr. Uratani, and Mr. Yano of Mitsubishi Precision Co., Ltd., who helped us to develop the vitreous surgery simulator.

6. REFERENCES

Mukai, N., M. Harada, , K. Muroi, T. Terada, T. Hikichi, and A. Yoshida, Vitreous surgery simulator System, *Japanese Journal of Medical Electronics and Biological Engineering*, 37, 282, 1999a.

Mukai, N., M. Harada, K. Muroi, and T. Terada, Ocular surgery simulator system, *Mitsubishi Denki Technical Report*, 73(11), 34-37, 1999b.

Neumann, P. F., L. L. Sadler, and J. Gieser, Virtual reality vitrectomy simulator, *Lecture Notes in Computer Science*, 1496, 910-917, 1998.

Suzuki, N., A. Hattori, T. Ezumi, T. Kumano, A. Ikemoto, Y. Adachi, and A. Takatsu, Development of virtual surgery system with sense of touch, *Transactions of the Virtual Reality Society of Japan*, 3(4), 237-243, 1998a.

Suzuki, N., A. Hattori, A. Takatsu, T. Kumano, A. Ikemoto, Y. Adachi, and A. Uchiyama, Virtual surgery system using deformable organ models and force feedback system with three fingers, *Lecture Notes in Computer Science*, 1496, 397-403, 1998b.

Takahashi, E., K. Hirota, and T. Kaneko, Surgical simulation of elastic organs, *Medical Imaging Technology*, 14(4), 479-480, 1996.